CHƯƠNG 10
LỚP LƯU TRỮ CỦA BIẾN
SỰ CHUYỂN KIỂU

CHƯƠNG 10
LỚP LƯU TRỮ CỦA BIẾN
SỰ CHUYỂN KIỂU

10.1 KHÁI NIỆM

Mỗi biến khi được sử dụng trong chương trình đều phải được khai báo, tuy nhiên biến có thể được khai báo ở nhiều chỗ trong chương trình, biến có thể được khai báo trong hàm, ngoài hàm..., mỗi chỗ như vậy sẽ làm cho biến có khả năng sử dụng khác nhau, từ đó hình thành nên các lớp lưu trữ biến.

CHƯƠNG 10
LỚP LƯU TRỮ CỦA BIẾN
SỰ CHUYỂN KIỂU

10.1 KHÁI NIỆM

Đối với C, dựa vào cách mà biến được lưu trữ và sử dụng, biến sẽ ở một trong các lớp lưu trữ khác nhau sau đây:

- Lớp biến tự động
- Lớp biến toàn cục và biến cục bộ
- Lớp biến tĩnh
- Lớp biến thanh ghi

Có hai đặc tính quan trọng của một biến: tầm sử dụng của biến và thời gian tồn tại của biến.

CHƯƠNG 10
LỚP LƯU TRỮ CỦA BIẾN
SỰ CHUYỂN KIỂU

10.1 KHÁI NIỆM

Tầm sử dụng của biến (scope) là nơi mà biến có thể được sử dụng trong các lệnh của chương trình. Do đặc tính này mà ta có hai lớp lưu trữ khác nhau là

- **lớp lưu trữ biến toàn cục** (global storage class)
- **lớp lưu trữ biến cục bộ** (local storage class).

CHƯƠNG 10
LỚP LƯU TRỮ CỦA BIẾN
SỰ CHUYỂN KIỂU

10.1 KHÁI NIỆM

Thời gian tồn tại của biến (time life) xác định rằng biến với giá trị đang tồn tại trong nó sẽ có ý nghĩa đến lúc nào. Sinh ra 2 lớp:

-lớp biến tự động (auto)

-lớp biến tĩnh (static)

CHƯƠNG 10
LỚP LƯU TRỮ CỦA BIẾN SỰ CHUYỂN KIỂU

10.1 KHÁI NIỆM

Lớp biến / Lớp biến	Tự động	Tĩnh
Toàn cục	(không kết hợp được)	Biến toàn cục tĩnh
Cục bộ	Biến cục bộ tự động (hay biến tự động)	Biến cục bộ tĩnh

10.2 BIẾN TOÀN CỤC VÀ BIẾN CỤC BỘ

10.2.1 Biến cục bộ

Biến cục bộ, còn gọi là biến tự động (auto), là các biến được khai báo ngay sau cặp dấu móc { và } (cặp dấu này như đã biết để bắt đầu cho một lệnh phức hoặc một thân hàm), hoặc là các biến được khai báo trong danh sách đối số của hàm.

CHƯƠNG 10
LỚP LƯU TRỮ CỦA BIẾN
SỰ CHUYỂN KIỂU

10.2 BIẾN TOÀN CỤC VÀ BIẾN CỤC BỘ

10.2.1 Biến cục bộ

Khi khai báo biến cục bộ ta có thể đặt hoặc không đặt từ khóa auto phía trước khai báo biến cục bộ theo cú pháp như sau:

[auto] kiểu danh_sách_tên_biến;

Ví dụ:

```
int tong (int n)
        {
                auto int i;          ...
        }
```

CHƯƠNG 10
LỚP LƯU TRỮ CỦA BIẾN
SỰ CHUYỂN KIỂU

10.2 BIẾN TOÀN CỤC VÀ BIẾN CỤC BỘ

10.2.1 Biến cục bộ

Khi khai báo biến cục bộ ta có thể đặt hoặc không đặt từ khóa auto phía trước khai báo biến cục bộ theo cú pháp như sau:

[auto] kiểu danh_sách_tên_biến;

Ví dụ:

```
int tong (int n)
        {
                auto int i;          ...
        }
```

CHƯƠNG 10
LỚP LƯU TRỮ CỦA BIẾN
SỰ CHUYỂN KIỂU

10.2 BIẾN TOÀN CỤC VÀ BIẾN CỤC BỘ

10.2.1 Biến cục bộ

Ví dụ :

Xét chương trình sắp xếp hai số, in ra kết quả theo thứ tự từ lớn tới nhỏ

10.2 BIẾN TOÀN CỤC VÀ BIẾN CỤC BỘ

10.2.1 Biến cục bộ

```
#include <stdio.h>
#include <conio.h>
main()
{
        auto int a, b;
        clrscr();
        printf ("Moi nhap hai so: ");
        scanf ("%d %d", &a, &b);
```

10.2 BIẾN TOÀN CỤC VÀ BIẾN CỤC BỘ

10.2.1 Biến cục bộ

```
        if (b> a)
        {
                auto int temp;
                temp = a;
                a = b;
                b = temp;
        }
        printf("Ket qua sap xep hai so: %d %d \n", a, b);
        getch();
}
```

10.2 BIẾN TOÀN CỤC VÀ BIẾN CỤC BỘ

10.2.2 Biến toàn cục

Biến toàn cục (global) hay còn gọi là biến ngoài là biến được khai báo ở bên ngoài tất cả các hàm. Biến này có thể được sử dụng để liên kết trị giữa các hàm khác nhau mà việc truyền theo tham số trở nên rắc rối và phức tạp. Các hàm sử dụng chung biến toàn cục có thể nằm trong cùng một tập tin hoặc có thể nằm trong các tập tin khác nhau.

10.2 BIẾN TOÀN CỤC VÀ BIẾN CỤC BỘ

10.2.2 Biến toàn cục

Ví dụ : Xét chương trình ví dụ sau:

```
#include <stdio.h>
#include <conio.h>
int a, b;
void swap(void);
main()
{       clrscr();
        printf ("Moi nhap hai so: ");          scanf ("%d %d", &a, &b);
        swap();
        printf ("Ket qua sap xep hai so: %d %d \n", a, b);
        getch();
}
```

10.2 BIẾN TOÀN CỤC VÀ BIẾN CỤC BỘ

10.2.2 Biến toàn cục

```
void swap(void)
{
        if (b> a)
        {
                auto int temp;
                temp = a;
                a = b;
                b = temp;
        }
}
```

10.2 BIẾN TOÀN CỤC VÀ BIẾN CỤC BỘ

10.2.2 Biến toàn cục

Như vậy, nếu có một biến toàn cục nào đó đã được khai báo trong một module của chương trình, và một hàm trong một module khác lại muốn sử dụng biến này để truyền trị, C đưa ra cú pháp sau đây:

extern kiểu tên_biến_toàn_cục;

Khai báo này được đặt đầu module chương trình chứa hàm sử dụng biến toàn cục.

10.2 BIẾN TOÀN CỤC VÀ BIẾN CỤC BỘ

10.2.2 Biến toàn cục

Tương tự cho hàm:

extern kiểu tên_hàm (danh_sách_khai_báo_đối_số);

Khai báo này thật sự chỉ là prototype của hàm thêm từ khóa **extern** phía trước.

Ví dụ 11.6 (trang 327-328)

CHƯƠNG 10
LỚP LƯU TRỮ CỦA BIẾN
SỰ CHUYỂN KIỂU

10.3 BIẾN TĨNH (STATIC)

Để khai báo biến tĩnh ta cần thêm từ khóa **static** trước khai báo biến bình thường, cú pháp như sau:

static kiểu danh_sách_tên_biến;

Biến toàn cục tĩnh là biến khai báo ngoài tất cả các hàm, trong một module chương trình nào đó và chỉ có ý nghĩa sử dụng bởi các hàm trong cùng module đó mà thôi. Các hàm trong các module khác của chương trình không thể sử dụng được các biến toàn cục dạng static như thế này.

CHƯƠNG 10
LỚP LƯU TRỮ CỦA BIẾN
SỰ CHUYỂN KIỂU

10.3 BIẾN TĨNH (STATIC)

Biến cục bộ tĩnh là các biến được khai báo trong hàm và chỉ có ý nghĩa sử dụng trong hàm có khai báo đó mà thôi.

Nhưng các biến cục bộ tĩnh khác với biến cục bộ (hay tự động) ở thời gian tồn tại, biến tĩnh tồn tại suốt trong bộ nhớ từ lúc nó được sử dụng lần đầu tiên cho đến khi kết thúc chương trình, và giá trị của chúng không hề mất đi khi ra khỏi hoặc trở vào hàm chứa nó.

10.3 BIẾN TĨNH (STATIC)

Ví dụ :

```
static int a;
main()
        {
                clrscr();
                ...
        }
int func(void)
        {                       static int b;
                ...
        }
```

10.3 BIẾN TĨNH (STATIC)

Ví dụ: Xét chương trình tính tổng

s = 1 +...+ n

dùng hàm trong đó có khai báo biến static.

10.3 BIẾN TĨNH (STATIC)

```
#include <stdio.h>
#include <conio.h>
int tong (int a);
main()
        {       int n, i, kq;
                clrscr();
                printf ("Nhap tri n: ");
                scanf ("%d", &n);
                for (i = 1; i <= n; i++)
                        kq = tong (i);
                printf ("Ket qua: %d", kq);
                getch();                }
```

CHƯƠNG 10
LỚP LƯU TRỮ CỦA BIẾN
SỰ CHUYỂN KIỂU

10.3 BIẾN TĨNH (STATIC)

```
int tong (int a)
                {
                        static int tam = 0;
                        tam += a;
                        return tam;
                }
```


Trong chương trình trên, trong hàm tong(), ta có khai báo một biến cục bộ tĩnh, biến **tam**, biến này chỉ được khởi động trị một lần đầu chương trình, trị 0, sau đó trị của biến này luôn được giữ lại cho lần sử dụng sau

10.3 BIẾN TĨNH (STATIC)

Ví dụ: Khởi động trị của biến static tam trong hàm tổng

```
void xoa (void)
        {
                int temp;
                if ( (temp = tong(0)) != 0 )
                        tong(-temp);
        }
```

Ví dụ 11.9: (332-333)

10.3 BIẾN TĨNH (STATIC)

Hàm được khai báo là static thì nó chỉ có thể được sử dụng trong module mà nó được khai báo và định nghĩa mà thôi. Cú pháp khai báo và định nghĩa hàm static như sau:


```
static kiểu tên_hàm (danh_sách_khai_báo_đối_số)
        {
                ...
        }
```

CHƯƠNG 10
LỚP LƯU TRỮ CỦA BIẾN
SỰ CHUYỂN KIỂU

10.4 BIẾN REGISTER

Bộ dịch C cho phép tận dụng các tài nguyên có sẵn của máy để tối ưu hóa chương trình, một trong các tối ưu này là C cho phép lập trình viên sử dụng một số thanh ghi của bộ vi xử lý để khai báo biến, biến này gọi là biến thanh ghi (register). Khai báo biến thanh ghi:

register kiểu danh_sách_tên_biến;

với **kiểu** là kiểu khai báo cho biến, kiểu này chỉ có thể là int, char, unsigned, long hoặc pointer

10.4 BIẾN REGISTER

Ví dụ:

```
register int i;
register char c;
register unsigned u;
resister long l;
register int *r;
register t;
```

CHƯƠNG 10
LỚP LƯU TRỮ CỦA BIẾN
SỰ CHUYỂN KIỂU

10.4 BIẾN REGISTER

Tầm sử dụng và thời gian tồn tại của các biến thanh ghi tương tự như các biến cục bộ, nhưng chúng được truy xuất nhanh hơn các biến cục bộ bình thường vì chúng chính là các thanh ghi của bộ vi xử lý.

Các biến thanh ghi thường được sử dụng làm các biến điều khiển trong các vòng lặp hoặc các biến phải truy xuất nhiều lần trong chương trình.

CHƯƠNG 10
LỚP LƯU TRỮ CỦA BIẾN
SỰ CHUYỂN KIỂU

10.5 KHỞI ĐỘNG TRỊ CHO BIẾN Ở CÁC LỚP

Đối với biến toàn cục hoặc biến tĩnh, ngay sau khi được khai báo, mỗi biến sẽ được C tự động gán trị là 0.
Trong khi đó biến tự động và biến thanh ghi sẽ có giá trị không xác định (gọi là trị rác).
trị bằng một biểu thức hằng.

CHƯƠNG 10
LỚP LƯU TRỮ CỦA BIẾN
SỰ CHUYỂN KIỂU

10.5 KHỞI ĐỘNG TRỊ CHO BIẾN Ở CÁC LỚP

Trong suốt quá trình chạy chương trình, biến toàn cục và biến tĩnh chỉ có thể được khởi động trị một lần, đó là lần đầu tiên mà khai báo biến đó được thực thi.

Biến toàn cục và biến tĩnh có thể được khởi động trị bằng một biểu thức hằng.

Biến tự động và biến thanh ghi có thể được khởi động trị bằng một biểu thức mà giá trị của biểu thức tới lúc đó đã xác định(có thể gọi hàm).

CHƯƠNG 10
LỚP LƯU TRỮ CỦA BIẾN
SỰ CHUYỂN KIỂU

10.5 KHỞI ĐỘNG TRỊ CHO BIẾN Ở CÁC LỚP

Việc khởi động cho các biến thuộc kiểu dữ kiện có cấu trúc như mảng (array), struct và union chỉ có thể thực hiện được đối với các biến toàn cục hoặc biến tĩnh mà thôi

10.6 SỰ CHUYỂN KIỂU

Khi thực hiện các phép toán số học hoặc luân lý, C luôn thực hiện sự chuyển kiểu tự động.

C còn cho phép lập trình viên thực hiện việc chuyển kiểu bắt buộc, ép kiểu (type casting). Cú pháp để ép kiểu một biến, hằng hay biểu thức:

(type) giá_trị

10.6 SỰ CHUYỂN KIỂU

Ví dụ:

Cho khai báo biến sau:

```
int a = 10, b = 3;
double d;
```

biểu thức nào cho kết quả đúng, giải thích?(xem lại thứ tự ưu tiên các phép toán của C)

a) `d = (double)(a/b);`

b) `d = (double)a/b;`

c) `d = a/(double)b;`

CHƯƠNG 10
LỚP LƯU TRỮ CỦA BIẾN
SỰ CHUYỂN KIỂU

10.7 ĐỊNH VỊ VÙNG NHỚ CHO CÁC LỚP LƯU TRỮ

Có hai cơ chế cơ bản giúp bộ dịch làm công việc này:

-Bộ dịch cần dùng một cách đúng đắn bảng biểu trưng để theo dõi các biến trong quá trình dịch.

-Bộ dịch cũng theo một sự phân chia bộ nhớ hệ thống, nó cẩn thận định vị bộ nhớ cho các biến dựa vào các đặc tính cụ thể, với các vùng nhớ xác định dành riêng cho các đối tượng đặc biệt.

CHƯƠNG 10
LỚP LƯU TRỮ CỦA BIẾN
SỰ CHUYỂN KIỂU

10.7 ĐỊNH VỊ VÙNG NHỚ CHO CÁC LỚP LƯU TRỮ

10.6.1 Bảng biểu trưng

Bộ dịch C theo dõi các biến trong một chương trình với một bảng biểu trưng.

Mỗi đầu vào của bảng biểu trưng cho biến chứa:

(1) tên của biến,

(2) kiểu của biến,

(3) vị trí trong bộ nhớ mà biến đó được định vị.

(4) một danh hiệu chỉ định khu vực mà trong đó biến được khai báo (tức tầm vực của biến đó).

10.7 ĐỊNH VỊ VÙNG NHỚ CHO CÁC LỚP LƯU TRỮ

10.6.1 Bảng biểu trưng

Ví dụ 11.18: Chương trình tính tốc độ mạng

```
#include <stdio.h>
int main()
{   int amount; int rate; int time; int hours; int minutes;
    int seconds;
    //  Nhập: số lượng byte và tốc độ truyền của mạng
   printf (“Có bao nhiêu byte dữ liệu được truyền?  ”);
   scanf (“%d”, &amount);
   printf (“Tốc độ truyền (bytes/giây)?  ”);
   scanf (“%d”, &rate);
```

CHƯƠNG 10
LỚP LƯU TRỮ CỦA BIẾN
SỰ CHUYỂN KIỂU

10.7 ĐỊNH VỊ VÙNG NHỚ CHO CÁC LỚP LƯU TRỮ

10.6.1 Bảng biểu trưng

Ví dụ 11.18: Chương trình tính tốc độ mạng

```
// Tính thời gian theo số giây
 time = amount / rate;`
 // Chuyển thời gian sang giờ, phút giây
 hours = time / 3600;// 3600 giây trong một giờ
 minutes = (time % 3600) / 60;// 60 giây trong một phút
 seconds = (time % 3600) % 60;// phần dư còn lại là giây
 // Xuất ra kết quả
 printf("Thời gian: %dh %dm %ds\n", hours, minutes,
seconds); }
```

CHƯƠNG 10
LỚP LƯU TRỮ CỦA BIẾN
SỰ CHUYỂN KIỂU

10.7 ĐỊNH VỊ VÙNG NHỚ CHO CÁC LỚP LƯU TRỮ

10.6.1 Bảng biểu trưng

Danh hiệu	Kiểu	Vị trí (offset)	Tầm vực
Amount	int	0	Main
Hours	int	-3	Main
Minutes	int	-4	Main
Rate	int	-1	Main
Seconds	int	-5	Main
Time	int	-2	Main

CHƯƠNG 10
LỚP LƯU TRỮ CỦA BIẾN
SỰ CHUYỂN KIỂU

10.7 ĐỊNH VỊ VÙNG NHỚ CHO CÁC LỚP LƯU TRỮ

10.6.2 Định vị vùng nhớ cho biến

Có hai vùng nhớ mà các biến C được định vị ở đó: vùng dữ liệu toàn cục (*global data section*) và ngăn xếp thực thi (*run-time stack*) (ngoài ra còn có biến thanh ghi).

- Vùng biến toàn cục là nơi chứa tất cả các biến toàn cục. Tổng quát hơn, nó cũng là nơi chứa các biến tĩnh.
- Vùng stack thực thi là nơi chứa các biến cục bộ (hay lớp biến lưu trữ tự động).

Vùng offset trong bảng biểu trưng cung cấp thông tin chính xác về vị trí trong bộ nhớ của các biến. Nó cho biết số ô nhớ tính từ địa chỉ nền của vùng nhớ chứa biến.

CHƯƠNG 10
LỚP LƯU TRỮ CỦA BIẾN
SỰ CHUYỂN KIỂU

BÀI TẬP CUỐI CHƯƠNG

1. Viết một hàm sao cho mỗi lần gọi hàm thì hàm sẽ trả về một trị số ngay sau trị trước đó trong dãy số Fibonaci.
2. Viết chương trình với các hàm tính các biểu thức sau đây: dùng và không dùng biến thanh ghi

$$t = \frac{1+...+n}{n!} - \frac{1+...+(n-1)}{(n-1)!} + \frac{1+...+(n-2)}{(n-2)!} - ... + \frac{1}{1!}$$

$$T = \frac{(1+...+n)+(1+...+(n-1))+(1+...+(n-2))+...+1!}{n!+(n-1)!+(n-2)!+...+1!}$$

BÀI TẬP CUỐI CHƯƠNG

3. Viết một chương trình gồm hai module: main.c và func.c, trong đó module main.c lưu hàm main() có các lệnh gọi nhập ba hệ số của tam thức bậc hai, kiểm tra in ấn; còn trong module func.c lưu các hàm cần thiết để giải phương trình bậc hai và biện luận tam thức bậc hai.

4. In ra màn hình các số nguyên tố từ 1 đến 1.000. Dùng biến thanh ghi và không dùng biến thanh ghi. Kiểm tra thời gian.